package com.twitter.product\_mixer.core.feature.featurestorev1

import com.twitter.ml.api.DataRecord

import com.twitter.ml.api.transform.FeatureRenameTransform

import com.twitter.ml.featurestore.lib.EntityId

import com.twitter.ml.featurestore.lib.dynamic.BaseGatedFeatures

import com.twitter.ml.featurestore.lib.feature.BoundFeature

import com.twitter.ml.featurestore.lib.feature.BoundFeatureSet

import com.twitter.ml.featurestore.lib.feature.TimelinesAggregationFrameworkFeatureGroup

import com.twitter.ml.featurestore.lib.feature.{Feature => FSv1Feature}

import com.twitter.product\_mixer.core.feature.ModelFeatureName

import com.twitter.product\_mixer.core.feature.datarecord.FeatureStoreDataRecordFeature

import com.twitter.product\_mixer.core.model.common.UniversalNoun

import com.twitter.product\_mixer.core.pipeline.PipelineQuery

import com.twitter.servo.util.{Gate => ServoGate}

import com.twitter.timelines.configapi.FSParam

import scala.reflect.ClassTag

/\*\*

\* The base trait for all feature store features on ProMix. This should not be constructed directly

\* and should instead be used through the other implementations below

\* @tparam Query Product Mixer Query Type

\* @tparam Input The input type the feature should be keyed on, this is same as Query for query

\* features and

\* @tparam FeatureStoreEntityId Feature Store Entity Type

\* @tparam Value The type of the value of this feature.

\*/

sealed trait BaseFeatureStoreV1Feature[

-Query <: PipelineQuery,

-Input,

FeatureStoreEntityId <: EntityId,

Value]

extends FeatureStoreDataRecordFeature[Input, Value]

with BaseGatedFeatures[Query] {

val fsv1Feature: FSv1Feature[FeatureStoreEntityId, Value]

val entity: FeatureStoreV1Entity[Query, Input, FeatureStoreEntityId]

val enabledParam: Option[FSParam[Boolean]]

override final lazy val gate: ServoGate[Query] = enabledParam

.map { param =>

new ServoGate[PipelineQuery] {

override def apply[U](query: U)(implicit asT: <:<[U, PipelineQuery]): Boolean = {

query.params(param)

}

}

}.getOrElse(ServoGate.True)

override final lazy val boundFeatureSet: BoundFeatureSet = new BoundFeatureSet(Set(boundFeature))

val boundFeature: BoundFeature[FeatureStoreEntityId, Value]

/\*\*

\* Since this trait is normally constructed inline, avoid the anonymous toString and use the bounded feature name.

\*/

override lazy val toString: String = boundFeature.name

}

/\*\*

\* A unitary (non-aggregate group) feature store feature in ProMix. This should be constructed using

\* [[FeatureStoreV1CandidateFeature]] or [[FeatureStoreV1QueryFeature]].

\* @tparam Query Product Mixer Query Type

\* @tparam Input The input type the feature should be keyed on, this is same as Query for query

\* features and

\* @tparam FeatureStoreEntityId Feature Store Entity Type

\* @tparam Value The type of the value of this feature.

\*/

sealed trait FeatureStoreV1Feature[

-Query <: PipelineQuery,

-Input,

FeatureStoreEntityId <: EntityId,

Value]

extends BaseFeatureStoreV1Feature[Query, Input, FeatureStoreEntityId, Value]

with ModelFeatureName {

val legacyName: Option[String]

val defaultValue: Option[Value]

override lazy val featureName: String = boundFeature.name

override final lazy val boundFeature = (legacyName, defaultValue) match {

case (Some(legacyName), Some(defaultValue)) =>

fsv1Feature.bind(entity.entity).withLegacyName(legacyName).withDefault(defaultValue)

case (Some(legacyName), \_) =>

fsv1Feature.bind(entity.entity).withLegacyName(legacyName)

case (\_, Some(defaultValue)) =>

fsv1Feature.bind(entity.entity).withDefault(defaultValue)

case \_ =>

fsv1Feature.bind(entity.entity)

}

def fromDataRecordValue(recordValue: boundFeature.feature.mfc.V): Value =

boundFeature.feature.mfc.fromDataRecordValue(recordValue)

}

/\*\*

\* A feature store aggregated group feature in ProMix. This should be constructed using

\* [[FeatureStoreV1CandidateFeatureGroup]] or [[FeatureStoreV1QueryFeatureGroup]].

\*

\* @tparam Query Product Mixer Query Type

\* @tparam Input The input type the feature should be keyed on, this is same as Query for query

\* features and

\* @tparam FeatureStoreEntityId Feature Store Entity Type

\*/

abstract class FeatureStoreV1FeatureGroup[

-Query <: PipelineQuery,

-Input,

FeatureStoreEntityId <: EntityId: ClassTag]

extends BaseFeatureStoreV1Feature[Query, Input, FeatureStoreEntityId, DataRecord] {

val keepLegacyNames: Boolean

val featureNameTransform: Option[FeatureRenameTransform]

val featureGroup: TimelinesAggregationFrameworkFeatureGroup[FeatureStoreEntityId]

override lazy val fsv1Feature: FSv1Feature[FeatureStoreEntityId, DataRecord] =

featureGroup.FeaturesAsDataRecord

override final lazy val boundFeature = (keepLegacyNames, featureNameTransform) match {

case (\_, Some(transform)) =>

fsv1Feature.bind(entity.entity).withLegacyIndividualFeatureNames(transform)

case (true, \_) =>

fsv1Feature.bind(entity.entity).keepLegacyNames

case \_ =>

fsv1Feature.bind(entity.entity)

}

}

sealed trait BaseFeatureStoreV1QueryFeature[

-Query <: PipelineQuery,

FeatureStoreEntityId <: EntityId,

Value]

extends BaseFeatureStoreV1Feature[Query, Query, FeatureStoreEntityId, Value] {

override val entity: FeatureStoreV1QueryEntity[Query, FeatureStoreEntityId]

}

trait FeatureStoreV1QueryFeature[-Query <: PipelineQuery, FeatureStoreEntityId <: EntityId, Value]

extends FeatureStoreV1Feature[Query, Query, FeatureStoreEntityId, Value]

with BaseFeatureStoreV1QueryFeature[Query, FeatureStoreEntityId, Value]

trait FeatureStoreV1QueryFeatureGroup[-Query <: PipelineQuery, FeatureStoreEntityId <: EntityId]

extends FeatureStoreV1FeatureGroup[Query, Query, FeatureStoreEntityId]

with BaseFeatureStoreV1QueryFeature[Query, FeatureStoreEntityId, DataRecord]

object FeatureStoreV1QueryFeature {

/\*\*

\* Query-based Feature Store backed feature

\* @param feature The underling feature store feature this represents.

\* @param \_entity The entity for binding the Feature Store features

\* @param \_legacyName Feature Store legacy name if required

\* @param \_defaultValue The default value to return for this feature if not hydrated.

\* @param \_enabledParam The Feature Switch Param to gate this feature, always enabled if none.

\* @tparam Query The Product Mixer query type this feature is keyed on.

\* @tparam FeatureStoreEntityId Feature Store Entity ID

\* @tparam Value The type of the value this feature contains.

\* @return Product Mixer Feature

\*/

def apply[Query <: PipelineQuery, FeatureStoreEntityId <: EntityId, Value](

feature: FSv1Feature[FeatureStoreEntityId, Value],

\_entity: FeatureStoreV1QueryEntity[Query, FeatureStoreEntityId],

\_legacyName: Option[String] = None,

\_defaultValue: Option[Value] = None,

\_enabledParam: Option[FSParam[Boolean]] = None

): FeatureStoreV1QueryFeature[Query, FeatureStoreEntityId, Value] =

new FeatureStoreV1QueryFeature[Query, FeatureStoreEntityId, Value] {

override val fsv1Feature: FSv1Feature[FeatureStoreEntityId, Value] = feature

override val entity: FeatureStoreV1QueryEntity[Query, FeatureStoreEntityId] = \_entity

override val legacyName: Option[String] = \_legacyName

override val defaultValue: Option[Value] = \_defaultValue

override val enabledParam: Option[FSParam[Boolean]] = \_enabledParam

}

}

object FeatureStoreV1QueryFeatureGroup {

/\*\*

\* Query-based Feature Store Aggregated group backed feature

\*

\* @param featureGroup The underling aggregation group feature this represents.

\* @param \_entity The entity for binding the Feature Store features

\* @param \_enabledParam The Feature Switch Param to gate this feature, always enabled if none.

\* @param \_keepLegacyNames Whether to keep the legacy names as is for the entire group

\* @param \_featureNameTransform Rename the entire group's legacy names using the [[FeatureRenameTransform]]

\* @tparam Query The Product Mixer query type this feature is keyed on.

\* @tparam FeatureStoreEntityId Feature Store Entity ID

\*

\* @return Product Mixer Feature

\*/

def apply[Query <: PipelineQuery, FeatureStoreEntityId <: EntityId: ClassTag](

\_featureGroup: TimelinesAggregationFrameworkFeatureGroup[FeatureStoreEntityId],

\_entity: FeatureStoreV1QueryEntity[Query, FeatureStoreEntityId],

\_enabledParam: Option[FSParam[Boolean]] = None,

\_keepLegacyNames: Boolean = false,

\_featureNameTransform: Option[FeatureRenameTransform] = None

): FeatureStoreV1QueryFeatureGroup[Query, FeatureStoreEntityId] =

new FeatureStoreV1QueryFeatureGroup[Query, FeatureStoreEntityId] {

override val entity: FeatureStoreV1QueryEntity[Query, FeatureStoreEntityId] = \_entity

override val featureGroup: TimelinesAggregationFrameworkFeatureGroup[

FeatureStoreEntityId

] = \_featureGroup

override val enabledParam: Option[FSParam[Boolean]] = \_enabledParam

override val keepLegacyNames: Boolean = \_keepLegacyNames

override val featureNameTransform: Option[FeatureRenameTransform] = \_featureNameTransform

}

}

sealed trait BaseFeatureStoreV1CandidateFeature[

-Query <: PipelineQuery,

-Input <: UniversalNoun[Any],

FeatureStoreEntityId <: EntityId,

Value]

extends BaseFeatureStoreV1Feature[Query, Input, FeatureStoreEntityId, Value] {

override val entity: FeatureStoreV1CandidateEntity[Query, Input, FeatureStoreEntityId]

}

trait FeatureStoreV1CandidateFeature[

-Query <: PipelineQuery,

-Input <: UniversalNoun[Any],

FeatureStoreEntityId <: EntityId,

Value]

extends FeatureStoreV1Feature[Query, Input, FeatureStoreEntityId, Value]

with BaseFeatureStoreV1CandidateFeature[Query, Input, FeatureStoreEntityId, Value]

trait FeatureStoreV1CandidateFeatureGroup[

-Query <: PipelineQuery,

-Input <: UniversalNoun[Any],

FeatureStoreEntityId <: EntityId]

extends FeatureStoreV1FeatureGroup[Query, Input, FeatureStoreEntityId]

with BaseFeatureStoreV1CandidateFeature[Query, Input, FeatureStoreEntityId, DataRecord]

object FeatureStoreV1CandidateFeature {

/\*\*

\* Candidate-based Feature Store backed feature

\* @param feature The underling feature store feature this represents.

\* @param \_entity The entity for binding the Feature Store features

\* @param \_legacyName Feature Store legacy name if required

\* @param \_defaultValue The default value to return for this feature if not hydrated.

\* @param \_enabledParam The Feature Switch Param to gate this feature, always enabled if none.

\* @tparam Query The Product Mixer query type this feature is keyed on.

\* @tparam FeatureStoreEntityId The feature store entity type

\* @tparam Input The type of the candidate this feature is keyed on

\* @tparam Value The type of value this feature contains.

\* @return Product Mixer Feature

\*/

def apply[

Query <: PipelineQuery,

Input <: UniversalNoun[Any],

FeatureStoreEntityId <: EntityId,

Value

](

feature: FSv1Feature[FeatureStoreEntityId, Value],

\_entity: FeatureStoreV1CandidateEntity[Query, Input, FeatureStoreEntityId],

\_legacyName: Option[String] = None,

\_defaultValue: Option[Value] = None,

\_enabledParam: Option[FSParam[Boolean]] = None

): FeatureStoreV1CandidateFeature[Query, Input, FeatureStoreEntityId, Value] =

new FeatureStoreV1CandidateFeature[Query, Input, FeatureStoreEntityId, Value] {

override val fsv1Feature: FSv1Feature[FeatureStoreEntityId, Value] = feature

override val entity: FeatureStoreV1CandidateEntity[Query, Input, FeatureStoreEntityId] =

\_entity

override val legacyName: Option[String] = \_legacyName

override val defaultValue: Option[Value] = \_defaultValue

override val enabledParam: Option[FSParam[Boolean]] = \_enabledParam

}

}

object FeatureStoreV1CandidateFeatureGroup {

/\*\*

\* Candidate-based Feature Store Aggregated group backed feature

\*

\* @param featureGroup The underling aggregation group feature this represents.

\* @param \_entity The entity for binding the Feature Store features

\* @param \_enabledParam The Feature Switch Param to gate this feature, always enabled if none.

\* @param \_keepLegacyNames Whether to keep the legacy names as is for the entire group

\* @param \_featureNameTransform Rename the entire group's legacy names using the [[FeatureRenameTransform]]

\* @tparam Query The Product Mixer query type this feature is keyed on.

\* @tparam Input The type of the candidate this feature is keyed on

\* @tparam FeatureStoreEntityId Feature Store Entity ID

\*

\* @return Product Mixer Feature

\*/

def apply[

Query <: PipelineQuery,

Input <: UniversalNoun[Any],

FeatureStoreEntityId <: EntityId: ClassTag,

](

\_featureGroup: TimelinesAggregationFrameworkFeatureGroup[FeatureStoreEntityId],

\_entity: FeatureStoreV1CandidateEntity[Query, Input, FeatureStoreEntityId],

\_enabledParam: Option[FSParam[Boolean]] = None,

\_keepLegacyNames: Boolean = false,

\_featureNameTransform: Option[FeatureRenameTransform] = None

): FeatureStoreV1CandidateFeatureGroup[Query, Input, FeatureStoreEntityId] =

new FeatureStoreV1CandidateFeatureGroup[Query, Input, FeatureStoreEntityId] {

override val entity: FeatureStoreV1CandidateEntity[Query, Input, FeatureStoreEntityId] =

\_entity

override val featureGroup: TimelinesAggregationFrameworkFeatureGroup[

FeatureStoreEntityId

] = \_featureGroup

override val enabledParam: Option[FSParam[Boolean]] = \_enabledParam

override val keepLegacyNames: Boolean = \_keepLegacyNames

override val featureNameTransform: Option[FeatureRenameTransform] = \_featureNameTransform

}

}